Real-time Heuristic Search in Dynamic Environments

Chao Chi Cheng and **Wheeler Ruml**
Department of Computer Science
University of New Hampshire
Durham, NH 03824 USA
cc1057 at wildcats.unh.edu, ruml at cs.unh.edu

Introduction

In dynamic environments such as cities, agents often do not have time to find a complete plan to reach a goal state. Planning in such environment requires an agent to update its plan frequently to respond to the changes around it. The setting of real-time heuristic search models on-line planning by requiring the agent to commit to its next action within a strict time limit. The time bound for planning is set to the time at which the actions to which the agent has already committed will end. In this way, planning and execution unfold in parallel. In this extended abstract, we summarize recent work (Cheng 2019) on real-time search algorithms that can tolerate a fully dynamic environment, in which action costs are not fully predictable. We present a combination of two previously-proposed methods and study its behavior experimentally. The results suggest that the new algorithm is significantly superior to the originals.

The real-time problem setting has been addressed by many previous algorithms, such as the paradigmatic LSS-LRTA* algorithm (Koenig and Sun 2009), but most of these only tolerate dynamic environments in which changes result in state transitions that are more expensive than originally anticipated. One exception is Partitioned Learning Real-Time A* (PLRTA*) (Cannon, Rose, and Ruml 2014), which also handles costs that decrease. The state space in dynamic environments usually includes time as a state variable, as being in the same position at a different time could have a different cost due to the dynamic obstacles. (As the time variable is a priori unbounded, this implies that the state space is infinite.) Since most environments include static elements that do not change over time, PLRTA* distinguishes between costs arising from static vs dynamic elements and ignores the time when learning the static cost-to-go. Therefore, the evaluation function of PLRTA* consists of four components — static cost-so-far $g_s(n)$, static cost-to-go $h_s(n)$, dynamic cost-so-far $g_d(n)$, and dynamic cost-to-go $h_d(n)$ — that are combined to form the f value:

$$f(n) = g_d(n) + g_s(n) + h_d(n) + h_s(n) \quad (1)$$

PLRTA* separates the learning process into two steps: static learning, and dynamic learning. For static learning, PLRTA* conflates all states that differ only in time into a single abstract state. Abstract states inherit the union of all the predecessors of their preimage states, so that backups can be performed properly. PLRTA* learns a single static heuristic value for each abstract state. For dynamic learning, PLRTA* performs the standard Dijkstra-style backup across the LSS, considering only costs arising from the dynamic elements of the environment. As presented by Cannon, Rose, and Ruml (2014), the algorithm commits to only one step along the selected path, and then replans using updated information. PLRTA* is guaranteed to reach a goal if there are no dynamic obstacles.

Dynamic-$\hat{f}$, proposed by Kiesel, Burns, and Ruml (2015), is a combination of two general enhancements to LSS-LRTA*. The first concerns the heuristic. Most heuristics are admissible, meaning that they are lower bounds on the actual cost. Kiesel, Burns, and Ruml (2015) suggest that real-time decision-making is best made on the basis of expected value, rather than lower bounds. They derive an estimate of expected cost $\hat{h}$ by learning online an estimate of how underestimating the provided h function is, and they use this to get an estimate of the expected cost $\hat{f}$:

$$\hat{h}(n) = h(n) + h_{error} \quad (2)$$

$$\hat{f}(n) = g(n) + \hat{h}(n) \quad (3)$$

The second enhancement concerns the treatment of lookahead. In real-time search, the lookahead size is strictly limited to ensure that an action can be selected within the real-time bound. This bound derives from the requirement that the agent never exhibit uncontrolled behavior: a new action must be ready by the time the currently executing action has completed. However, if the agent commits to multiple actions after a single planning phase, this means a larger real-time bound can be used for the next iteration. By dynamically adjusting lookahead to fill the available time in this way, action selection can be more informed, leading to better performance.

New Algorithms

In this work, we propose two new algorithms, $\hat{f}$-PLRTA* and Dyn-$\hat{f}$-PLRTA*, that enhance PLRTA* with the $\hat{f}$ and dynamic lookahead techniques from Dynamic-$\hat{f}$.

As in PLRTA*, $\hat{f}$-PLRTA* copes with dynamic environments by learning the static cost-to-go and dynamic cost-to-go separately. The central difference between $\hat{f}$-PLRTA*

and PLRTA* is that $\hat{f}$-PLRTA*considers the heuristic error when sorting the open list:

$$\hat{f}(n) = g_d(n) + g_s(n) + h_d(n) + h_s(n) + h_{error}(n) \quad (4)$$

$\hat{f}$-PLRTA* uses a best-first search on $\hat{f}$ to generate the local search space. Then it selects the node with the lowest $\hat{f}$ value as the next target state for the agent. Ties are broken in favor of the state with the highest time. To estimate the one-step heuristic error ϵ, we averaged the difference of the $f_{static} = g_s + h_s$ value of a parent and its best child — see Cheng (2019) for details. Although there is of course also error in the dynamic heuristic values, it is very hard to predict due to the inherent unpredictability of dynamic obstacles. Therefore, we only account for the static heuristic error.

Dyn-$\hat{f}$-PLRTA* adds the dynamic lookahead technique to $\hat{f}$-PLRTA*. After the lookahead phase, the basic version of Dyn-$\hat{f}$-PLRTA* commits to all the actions along the path from the current state to the target frontier node. To take into account the changing predictions of the future locations of dynamic obstacles, Dyn-$\hat{f}$-PLRTA* checks the cost of the remaining committed actions at each time step if the new cost of the path is now greater than the original cost times a factor (1.1 in our experiments). If yes, then the previously-committed actions are abandoned and the real-time search is restarted with a time bound of one action duration.

In preliminary experiments, we found that dynamic lookahead actually performed worse than committing to a single action at a time. Other have also reported this (Luštrek and Bulitko 2006; Kiesel, Burns, and Ruml 2015). This makes sense, as committing to only a single step ensures lookahead from every state the agent visits. In order to allow increasing the amount of planning while ensuring some lookahead, we use a conservative approach that constrains the number of committed actions to be at most one more than in the previous iteration.

Theorem 1 *In a finite state space with an admissible h and no dynamic obstacles or dead ends, Dyn-$\hat{f}$-PLRTA*is complete.*

Proof: With no dynamic obstacles, g_d and h_d will remain 0. Even if time is a state variable, h_s applies to abstract states, effecitvely removing it from consideration. Dyn-$\hat{f}$-PLRTA* will therefore be equivalent to Dynamic-$\hat{f}$, which is complete. □

Experimental Results

To evaluate the practical performance of these new methods, we tested them on a grid pathfinding problem with moving obstacles. The state space is $\{x, y, t\}$, where x and y represent position and the t represents time. The agent has 9 actions: 8-way movement plus wait. Dynamic obstacles move randomly with various speed and direction. The movements of dynamic obstacles are unpredictable to the agent, so the agent needs to predict the possible future states of a dynamic obstacle when planning. This is done by linear extrapolation from the two most recent time steps.

We ran our experiment on a 256 by 256 city map from (Sturtevant 2012). Each test instance has a different start and goal pair that is randomly generated and at least 128 away from each other in Euclidean distance. For every start and goal pair we have sub-instances that have different numbers of randomly generated dynamic obstacles.

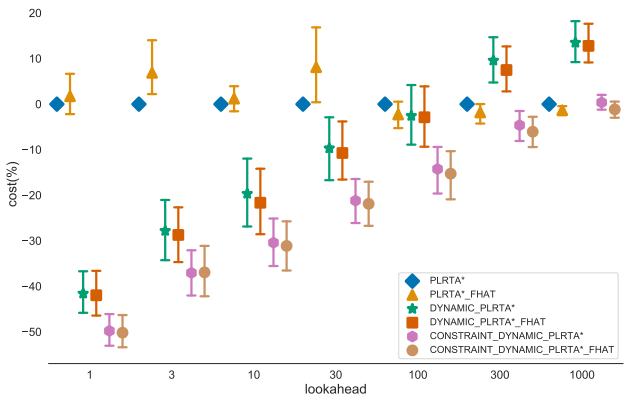

Figure 1: Comparison of Different Algorithms on City Map

Figure 1 presents the experimental results, shown as a relative percentage of the difference between the algorithm and PLRTA* (lower is better). One-step $\hat{f}$-PLRTA* does not show any improvement over the original PLRTA*. However, adding conservative dynamic lookahead strongly improves performance, with a stronger improvement for shorter lookaheads. (For large lookaheads, all algorithms will converge to optimal offline A*.) In the context of dynamic lookahead, using $\hat{f}$ does not seem to make much difference. The result also suggests the new constrained dynamic lookahead is a clear improvement over the original dynamic lookahead.

We also tested our algorithms on additional instances with similar results. For complete details, refer to Cheng (2019).

References

Cannon, J.; Rose, K.; and Ruml, W. 2014. Real-time motion planning with dynamic obstacles. *Ai Communications* 27:345–362.

Cheng, C. C. 2019. Real-time search in dynamic worlds. B.S. Thesis, University of New Hampshire.

Kiesel, S.; Burns, E.; and Ruml, W. 2015. Achieving goals quickly using real-time search: Experimental results in video games. *Journal of Artificial Intelligence Research* 54:123–158.

Koenig, S., and Sun, X. 2009. Comparing real-time and incremental heuristic search for real-time situated agents. *Autonomous Agents and Multi-Agent Systems* 18(3):313–341.

Luštrek, M., and Bulitko, V. 2006. Lookahead pathology in real-time path-finding. In *Proceedings of the National Conference on Artificial Intelligence (AAAI), Workshop on Learning For Search*, 108–114.

Sturtevant, N. 2012. Benchmarks for grid-based pathfinding. *Transactions on Computational Intelligence and AI in Games* 4(2):144 – 148.